STAT 1291: Data Science

Lecture 18 - Statistical modeling II: Machine learning

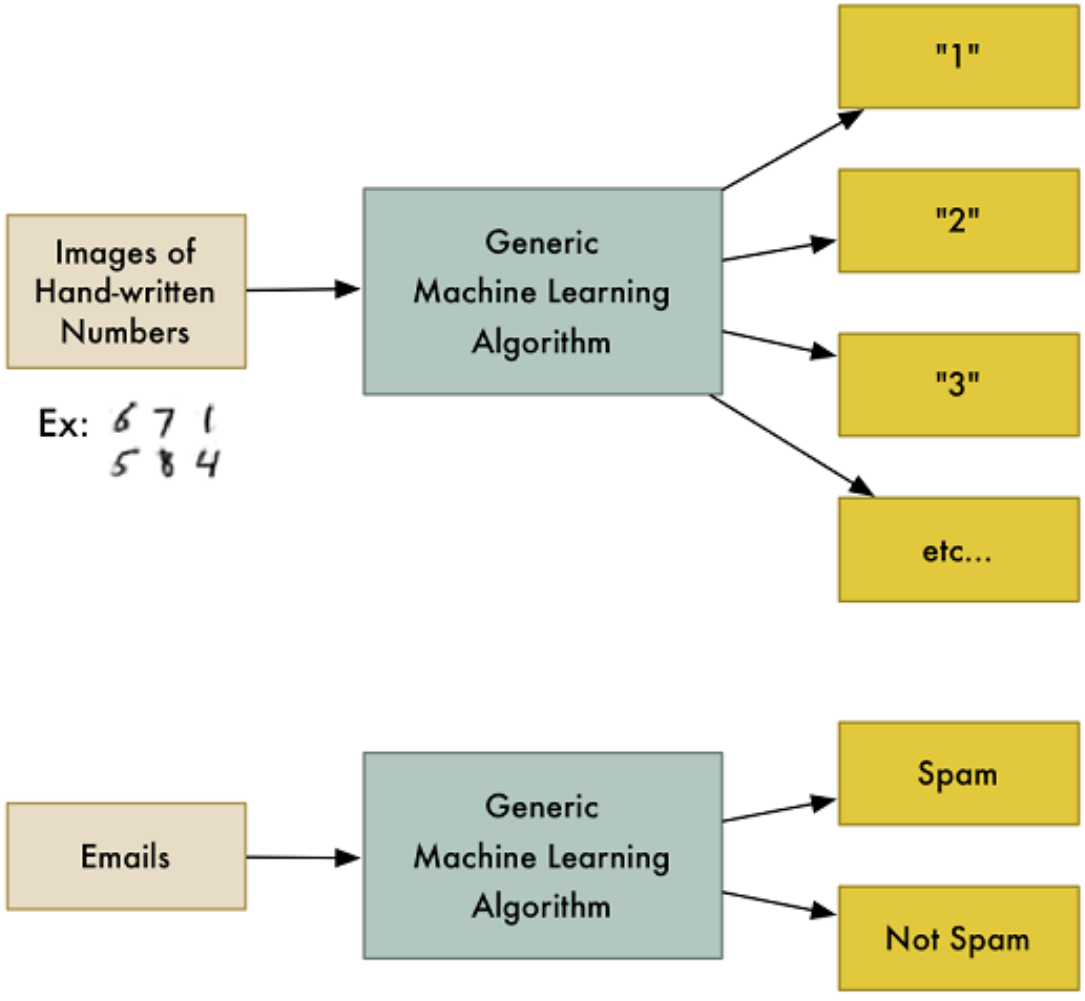
Sungkyu Jung

Where are we?

- *data visualization*
- *data wrangling*
- *professional ethics*
- *statistical foundation*
- *Statistical modeling: Regression*
- *Cause and effect: Causality and confounding*
- More statistical modeling: Machine learning

Machine learning

A popular view of ML algorithms. ML can be applied to **any** data, if they are in the right form



Machine learning

Machine learning algorithms are largely used to predict, classify, or cluster.

- *Prediction* and *classification* are examples of *supervised learning*
- *clustering* is an example of *unsupervise learning*.

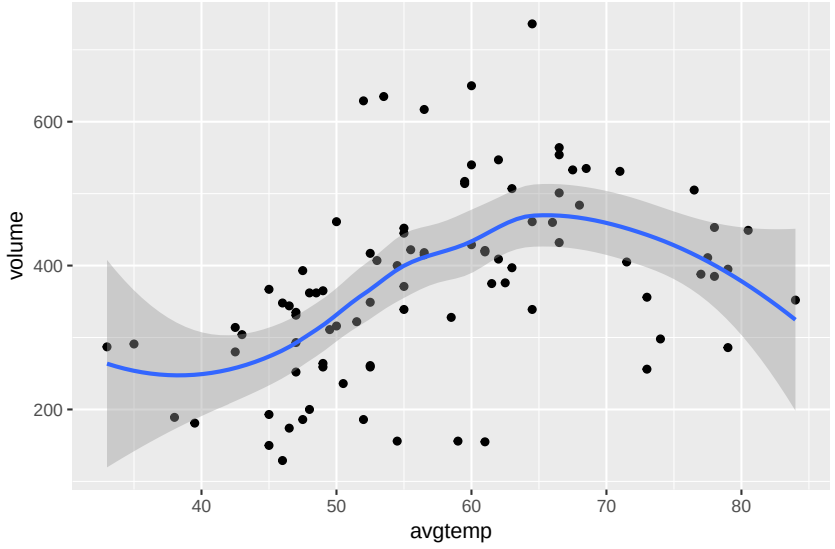
Put another way, supervised learning is concerned with problems that have a response variable and unsupervised learning is concerned with problems without a response variable.

Supervised learning

1. Supervised learning—Prediction (of numeric response)

Predict the ridership (`volume`) using other information, in `RailTrail` data:

```
RailTrail %>% ggplot(aes(x = avgtemp, y = volume)) +
  geom_point() +
  geom_smooth(method = 'loess')
```

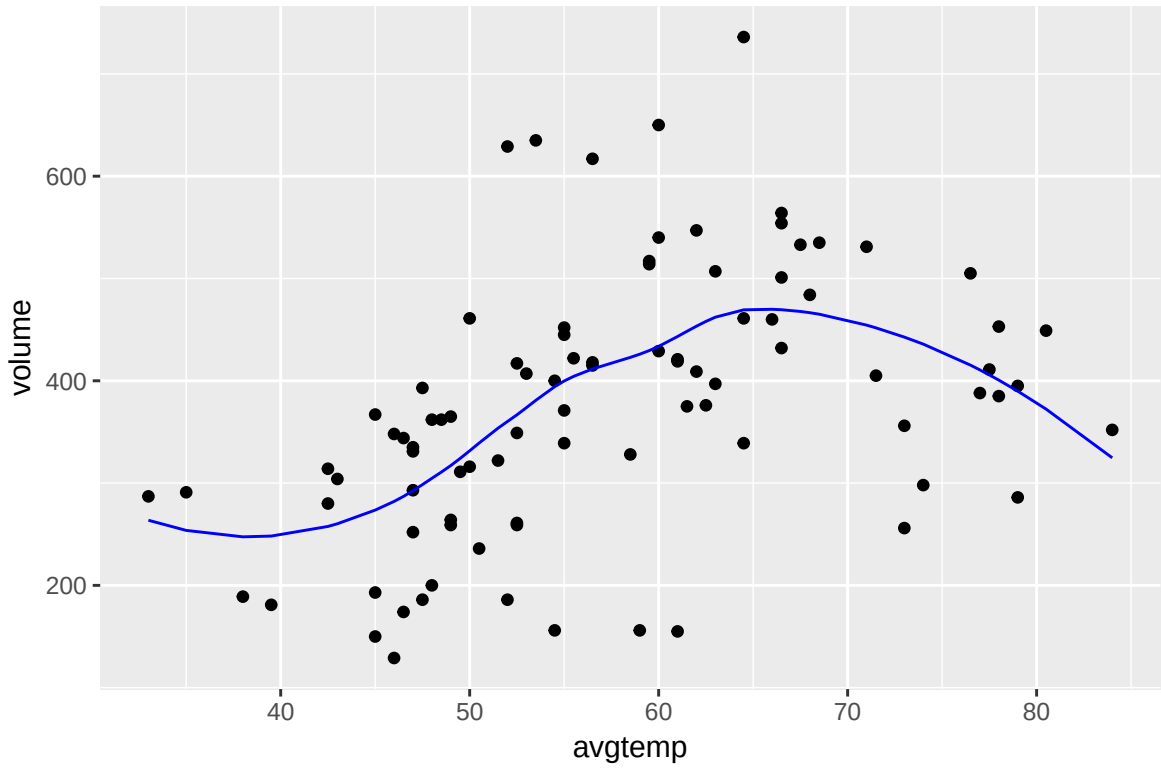


- Response is `volume`, which is "supervising" the model fit.
- The scatterplot smoother (in this case, "loess") is predicting `volume`.

```
RailTrail.fit <- RailTrail %>%
  mutate(
    loess.fit = loess(volume ~ avgtemp, data = .)$fitted)
head(select(RailTrail.fit, volume, loess.fit, avgtemp))
```

```
##   volume loess.fit avgtemp
## 1    501  469.4541    66.5
## 2    419  443.2781    61.0
## 3    397  462.1633    63.0
## 4    385  400.6197    78.0
## 5    200  304.6494    48.0
## 6    375  448.3692    61.5
```

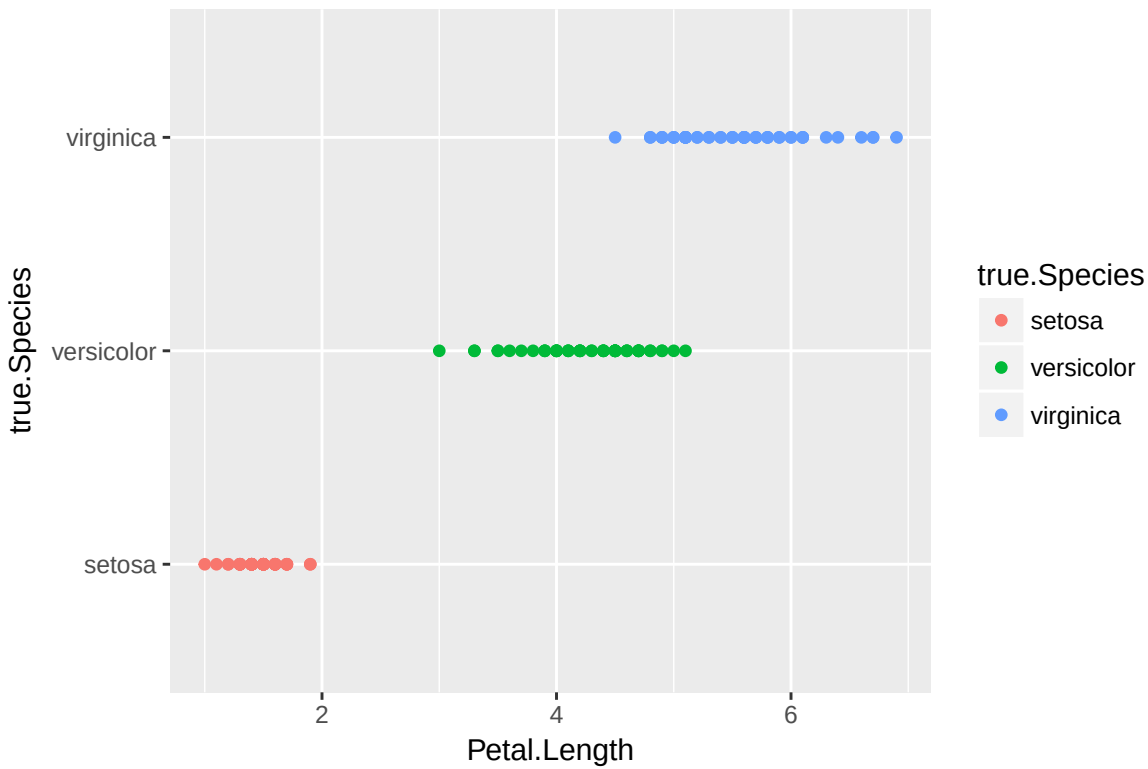
```
RailTrail.fit %>%
  ggplot() +
  geom_point(aes(x = avgtemp, y = volume)) +
  geom_line(aes(x = avgtemp, y = loess.fit), color = "blue"
```



2. Supervised learning—Classification

- The Iris data set: `https://en.wikipedia.org/wiki/Iris_flower_data_set`

```
iris %>% mutate(true.Species = Species) %>%
  ggplot(aes(x = Petal.Length, y = true.Species, color = tr
```

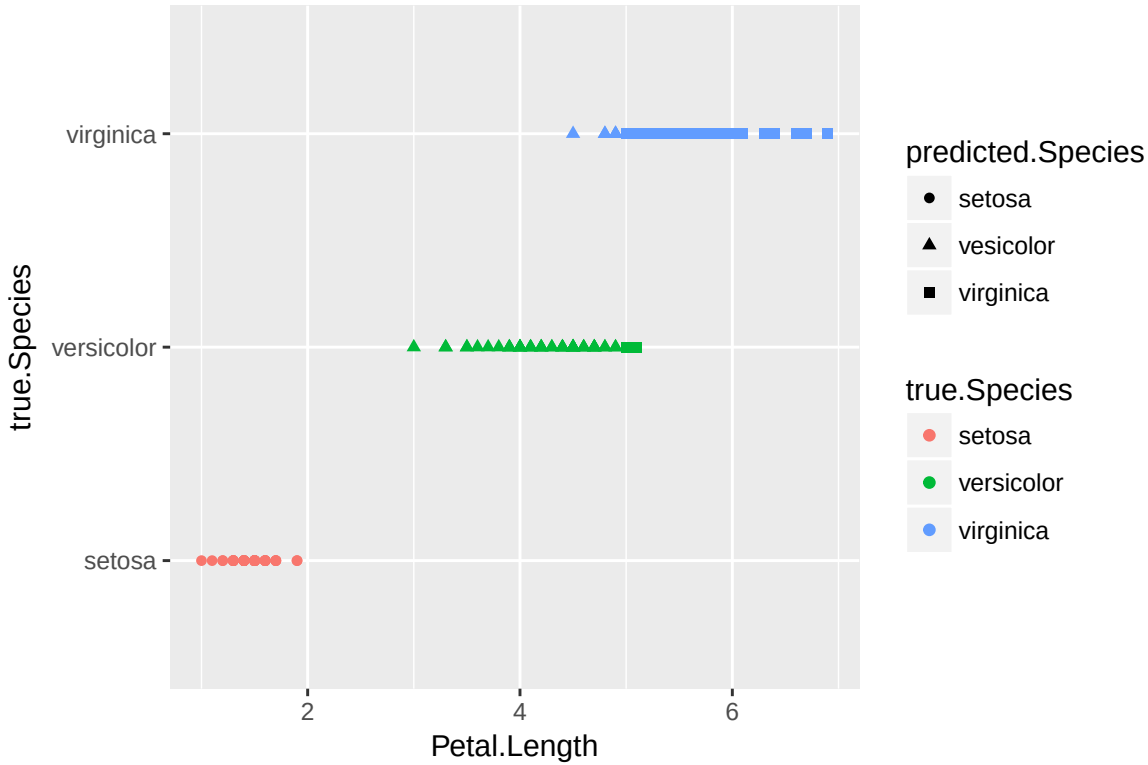


Can you classify these flowers (using the measurement `Petal.Length`) into one of the species?

- A crude answer:
  - Petal.Length $<$ 2 : setosa
  - 2 $<$ Petal.Length $<$5 : versicolor
  - Petal.Length $>$ 5 : virginica

```
iris.crude <- iris %>% mutate(true.Species = Species) %>%
  mutate(
    predicted.Species =
      ifelse(Petal.Length < 2, "setosa",
             ifelse(Petal.Length < 5, "vesicolor",
                    "virginica"))
    )
```

```
iris.crude %>%
  ggplot(aes(x = Petal.Length, y = true.Species,
             color = true.Species, shape = predicted.Specie
  geom_point()
```



- A crude eyeballing gives a good classification.
- "Supervising" response variable is `true.Species`.

Classification is just a special prediction

- Two variables `true.Species` and `predicted.class` have the same set of values "setosa, versicolor and virginica".

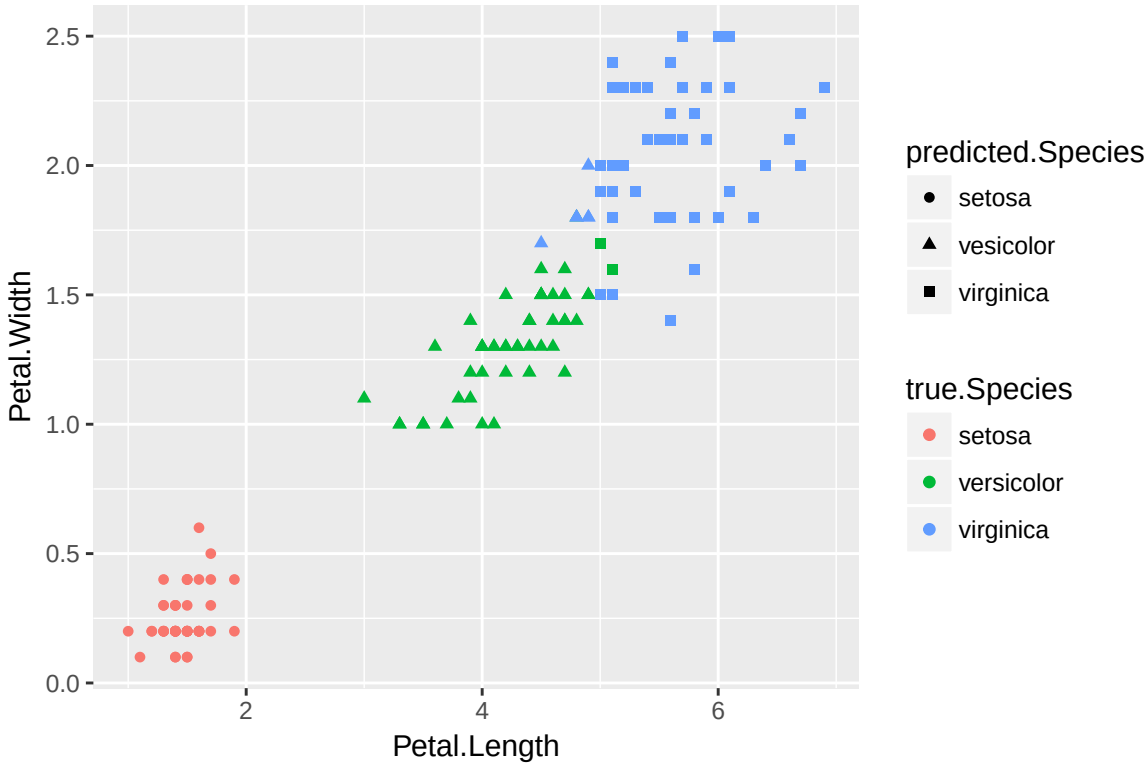
```
set.seed(13)
sample_n(select(iris.crude, true.Species, predicted.Species
```

```
##     true.Species predicted.Species Petal.Length
## 107    virginica         vesicolor          4.5
## 37        setosa            setosa          1.3
## 58    versicolor         vesicolor          3.3
## 14        setosa            setosa          1.1
## 141    virginica         virginica          5.6
```

- This is just predicting `true.Species` by `predicted.Species`.

We can classify better when more variables are available.

```
iris.crude %>% ggplot(aes(x = Petal.Length, y = Petal.Width
```



- The scatterplot above was created using only 2 variables.
- How to use all 4 variables in classification?
  - Visualization itself is limited in answering this question
  - We will use model-based, or algorithm-based methods in classification
- Visualization will be still useful
  - in exploratory analysis (deciding which method to use);
  - in conveying the analysis result.
- Classification will be discussed in more detail in the next lecture

Unsupervised learning

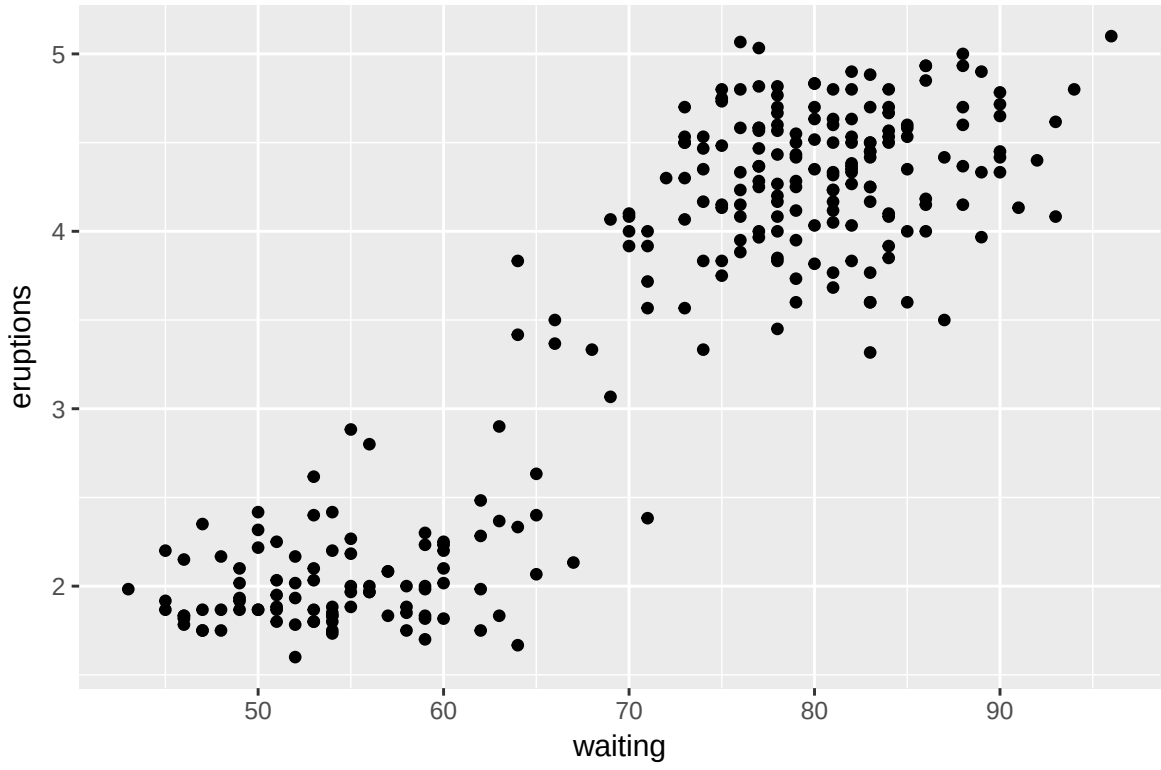
1. Unsupervised learning—clustering

Example: Waiting time between eruptions and the duration of the eruption for the Old Faithful geyser in Yellowstone National Park, Wyoming, USA.

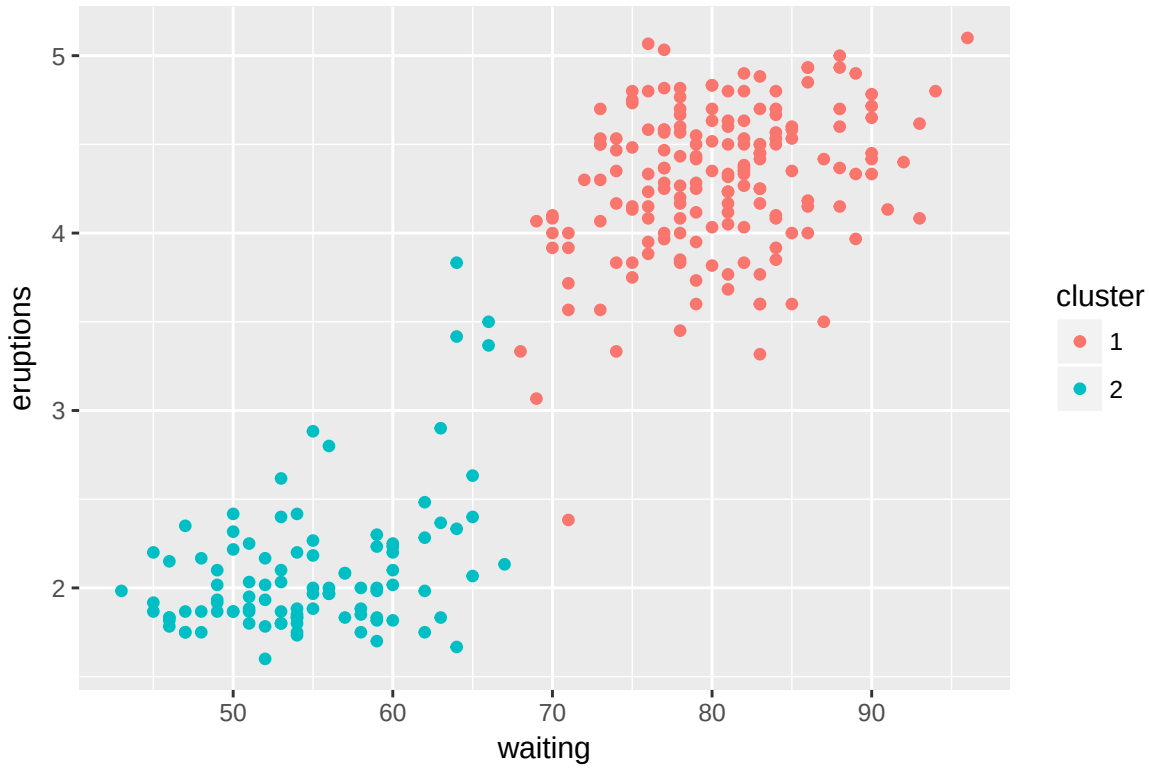
The `faithful` dataset contains the eruption times in minutes and the waiting time to next eruption (in minutes).

```
head(faithful)
```

```
##   eruptions waiting
## 1     3.600      79
## 2     1.800      54
## 3     3.333      74
## 4     2.283      62
## 5     4.533      85
## 6     2.883      55
```



Do you see two “clusters”?


eruptions
5
4
3
2
50
60
70
80
90
waiting
cluster
1
2

Clustering methods

There are a number of clustering methods available

Many are algorithm-based:

1. Hierarchical clustering (Agglomerative vs divisive)
2. Partitioning methods (K-means, K-medoids)

Some are model-based:

1. Gaussian mixture

K-means

- K-means clustering is an algorithm-based approach to find k clusters in the data
- Use "heuristic" updates to make clusters as far as possible, and each of clusters as tight as possible
- Why heruristic? The number of different clustering of $n = 25$ observations into $k = 4$ clusters exceed 10^{13}.
- The number sharply increases as n and k increase.
- It calls for more efficient algorithm (may not be optimal but reasonably good sub-optimal solutions): K-means algorithm

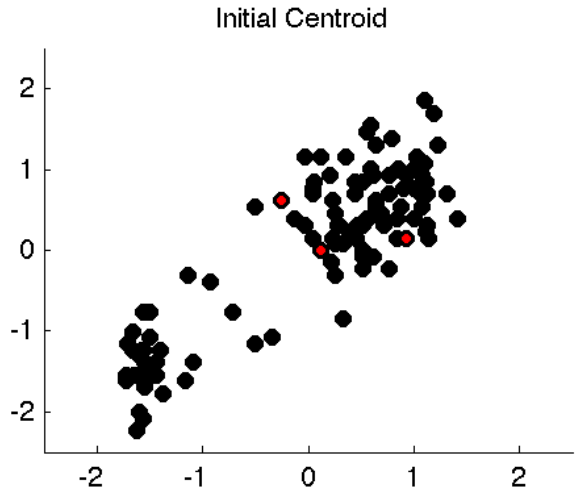
K-means algorithm

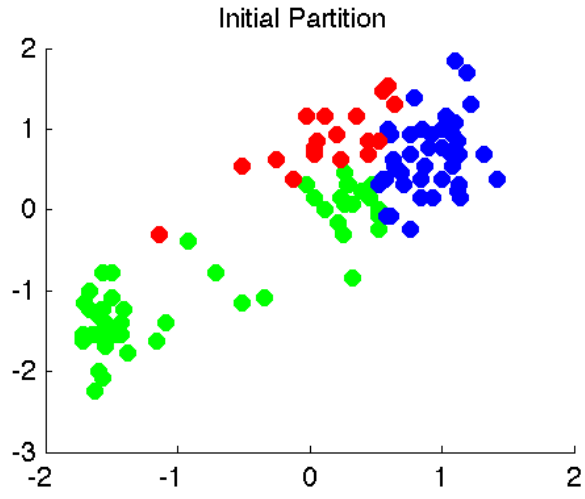
The algorithmic iteration begins with an initial guess for K cluster means $m_1, \ldots, m_K$.

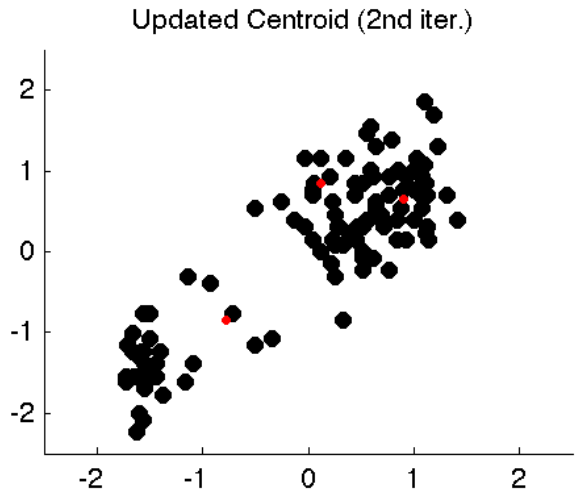
1. Update cluster: For each observation, cluster it to the closest mean m_j
2. Update K-means: For each cluster, update m_j by the new average of points in cluster j.
3. Iterate Steps 1 and 2.

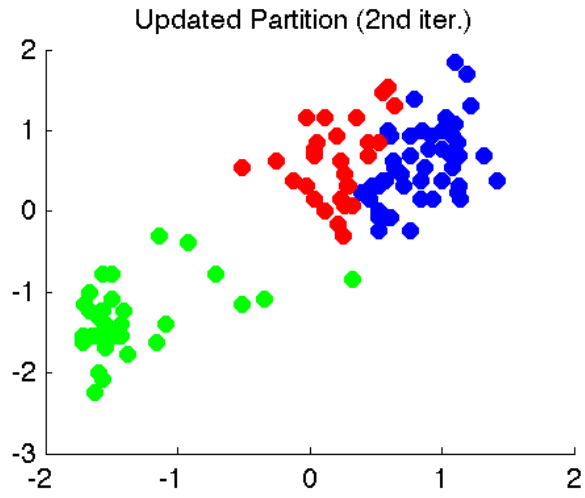
K-means example

First two iterations of the algorithm



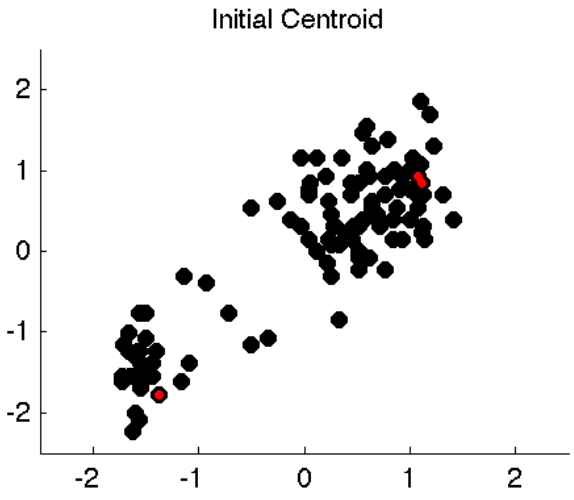


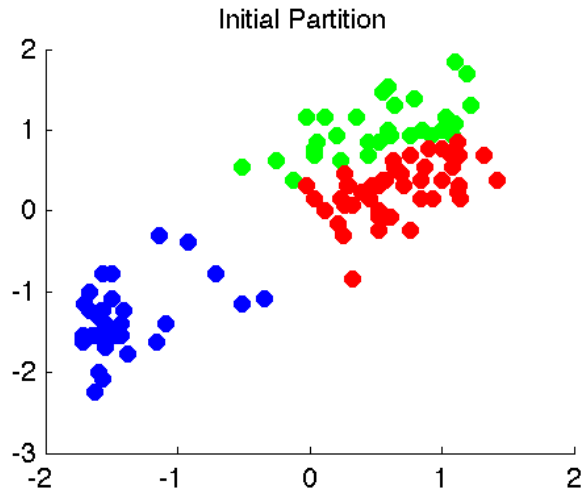


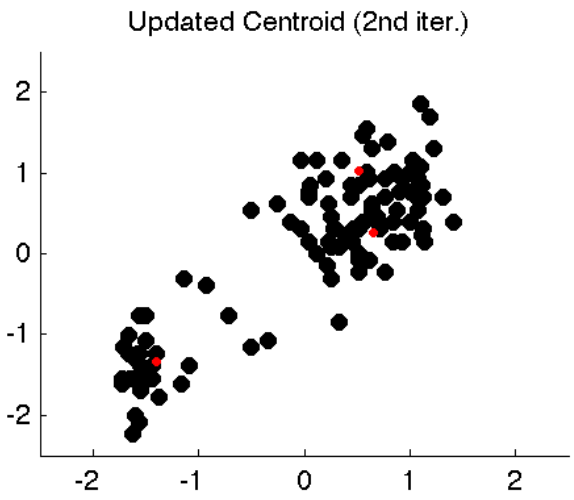


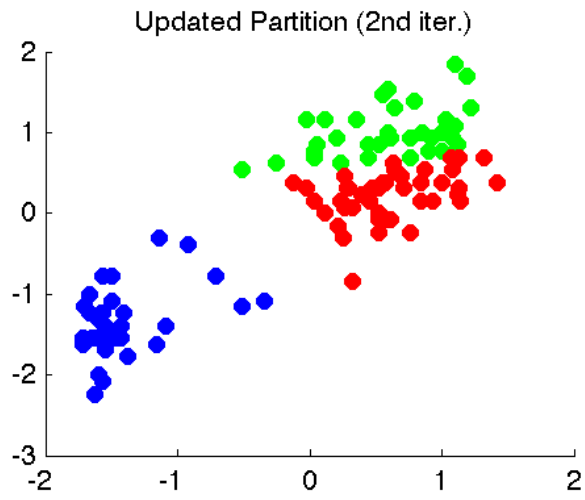
K-means example

First two iterations of the algorithm (with different initial)



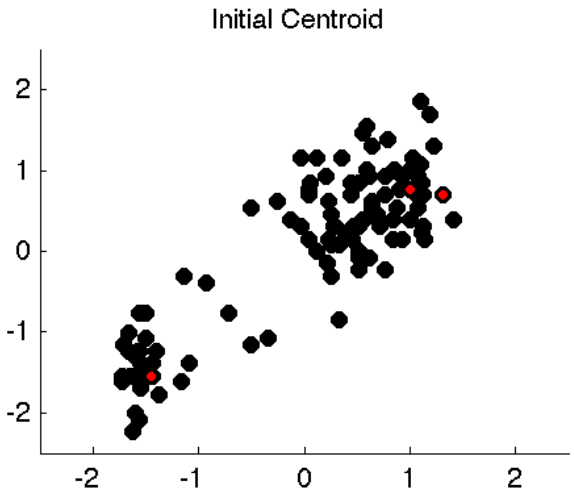


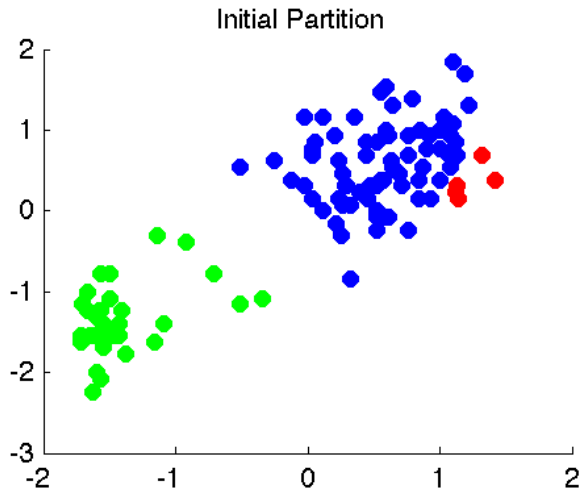


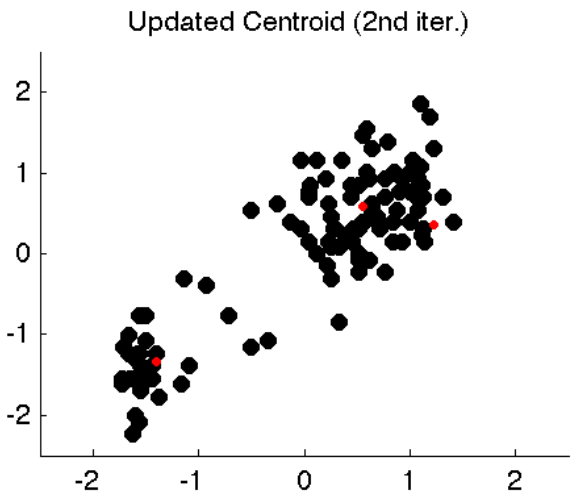


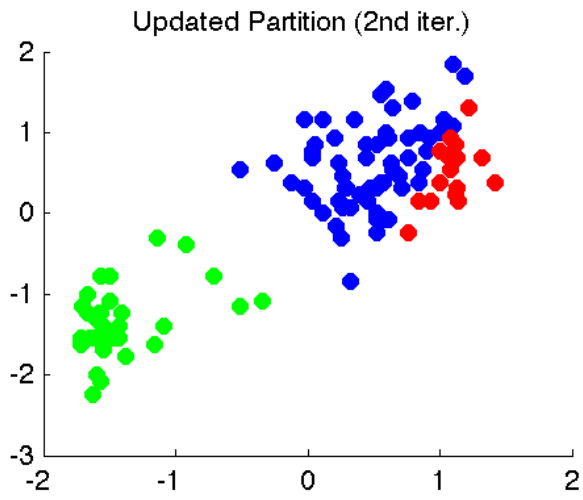
K-means example

First two iterations of the algorithm (with another different initial)









- R function `kmeans()` can be used for this task.

```
faithful.clustered <-
  faithful %>%
  mutate(cluster =
           factor(kmeans(x = ., centers = 2)$cluster)
         )

faithful.clustered %>%
  ggplot(aes(y = eruptions, x = waiting)) +
  geom_point(aes(color = cluster))
```

2. Unsupervised learning—dimension reduction

Example 1. The Iris data

The iris data has four numeric variables; an observation is a vector of length 4.

- Dimension of the problem = dimension of the vector space = number of variables
- For dimension 2 (two variables), a scatterplot is the best visualization method.
- For dimension 3 (three variables), one may use an interactive 3D scatterplot.

For interactive graphs, including 3D scatterplot, using Plot.ly is recommended.

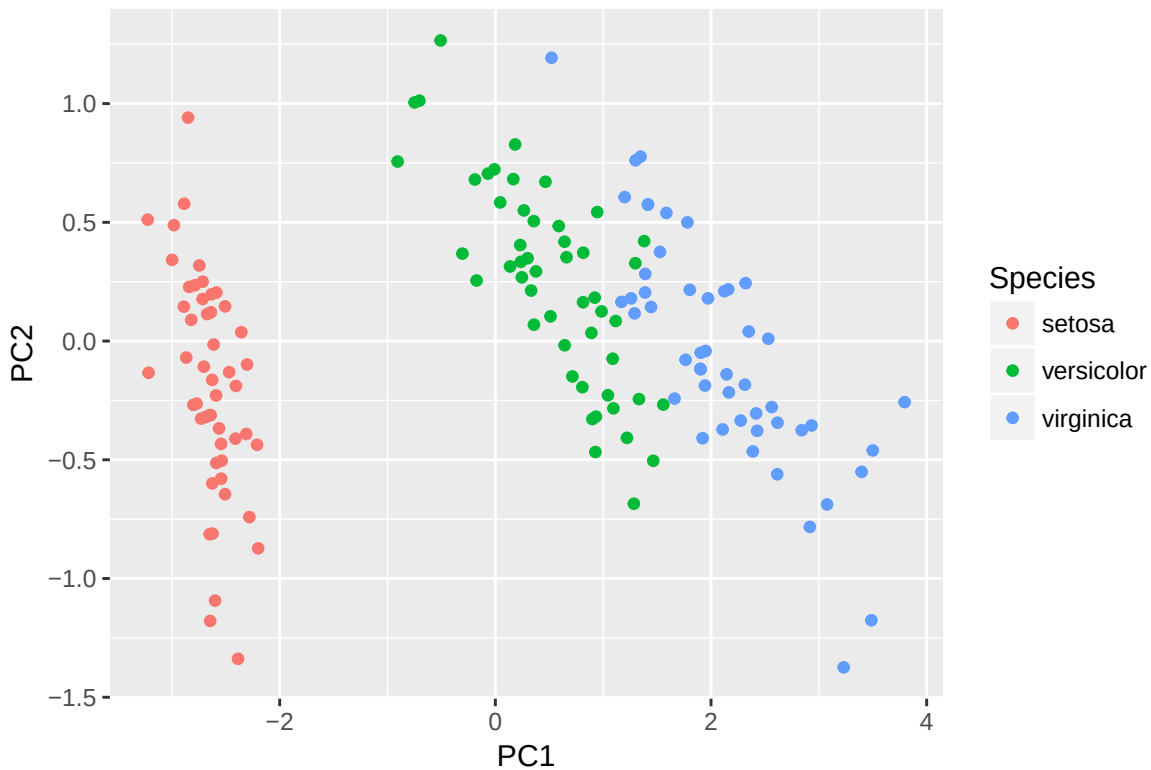
- Plot.ly specializes in online dynamic data visualizations, and in particular, the ability to translate code to generate data graphics between R, Python, and other data software tools.
- What makes plotly especially attractive is that it can convert any ggplot2 object into a plotly object using the `ggplotly()` function.
- See MDSR Section 11.1.2 for an introduction to `ggplotly()`.

```
# For demo
library(plotly)
plot_ly(iris,
        x = ~Petal.Length,
        y = ~Sepal.Length,
        z = ~Petal.Width,
        color = ~Species) %>%
  add_markers() %>%
  layout(scene =
           list(xaxis = list(title = 'Petal.Length'),
                yaxis = list(title = 'Sepal.Length'),
                zaxis = list(title = 'Petal.Width')))
```

- Each snapshot of the 3D scatterplot is a 2D scatterplot.
- Different snapshots are given by different viewpoint.
- Some are useful; some are not.
- A dimension reduction (from 3 to 2, in this case) is to find a "right" viewpoint.
- This is achieved by Principal Component Analysis, which amounts to a rotation of the coordinate axes so that more of the variability can be explained using just a few variables.

Principal Component Analysis

```
pc.iris <- iris %>% select(-Species) %>% prcomp()
pc.iris$x %>%
  as_tibble() %>%
  bind_cols(iris) %>%
  ggplot(aes(x = `PC1`, y = `PC2`, color = Species)) + geom
```



Principal Component Analysis (PCA)

- PCA is an example of *linear* dimension reduction.
- PCA creates new variables, ordered by its importance.
- A new variable is a linear function of old variables:

E.g. the first new variable, named the first principal component, is

$$\begin{aligned} \text{PC}_1 =& 0.36\text{Sepal.Length} - 0.08\text{Sepal.Width} && (1)\\ &+ 0.86\text{Petal.Length} + 0.36\text{Petal.Width}. && (2) \end{aligned}$$

```
pc.iris$rotation[,1]
```

```
## Sepal.Length  Sepal.Width Petal.Length  Petal.Width
##   0.36138659  -0.08452251   0.85667061   0.35828920
```

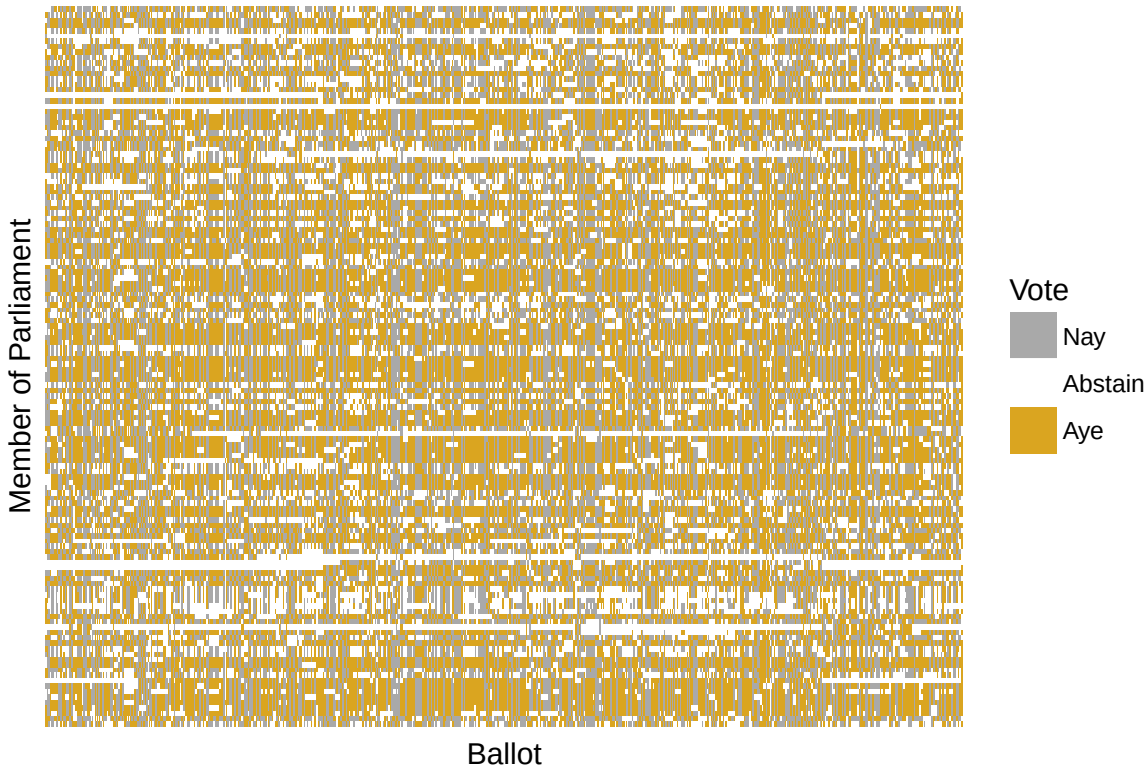
Example 2. Votes from Scottish Parliament

Votes recorded on each ballot by each member of the Scottish Parliament in 2008.

Legislators often vote together in pre-organized blocks, and thus the pattern of "ayes" and "nays" on particular ballots may indicate which members are affiliated (i.e., members of the same political party)

```
## 'data.frame':    103582 obs. of  3 variables:
##  $ bill: Factor w/ 773 levels "S1M-1","S1M-1007.1",..: 1
##  $ name: chr  "Canavan, Dennis" "Canavan, Dennis" "Canav
##  $ vote: int  1 1 1 -1 -1 -1 -1 1 -1 -1 ...
```

- Can the members of the parliament be grouped, according to their voting patterns?
- Can the bills be grouped into several categories, according to the voting patterns?



Suppose that you wish to cluster the members into groups. It helps to transforme the narrow data into wide format (as shown above), so that each case (row) corresponds to a member.

Use `tidyr::spread()` to achieve that.

```
wide.Votes <- Votes %>%
  spread(key = bill, value = vote) %>%
  as_tibble()
# wide.Votes
```

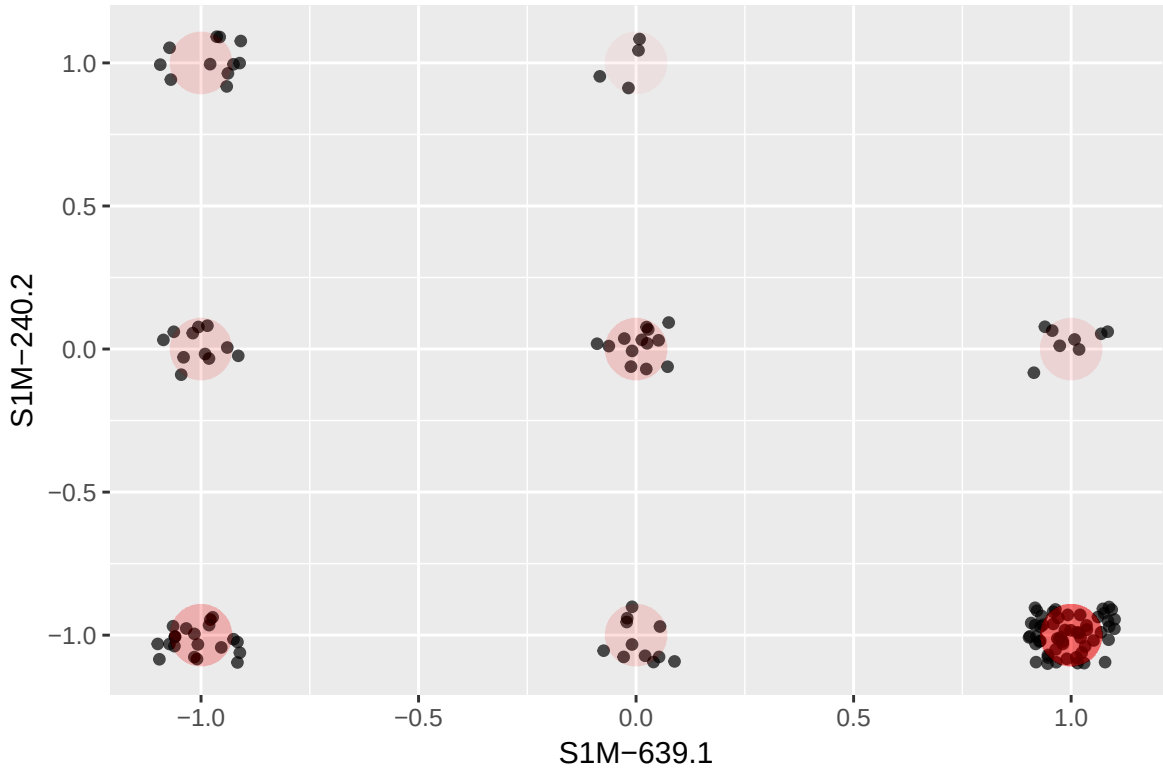
In the wide data, there are 134 rows (members) and 774 variables (bills).

For 2 variable (take `S1M-240.2` and `S1M-639.1`), we can take a look at the joint distribution. Recall that "Nay","Abstain","Aye" are coded as -1, 0, 1.

```
wide.Votes %>% select(c("S1M-240.2", "S1M-639.1")) %>%
  table()
```

```
##             S1M-639.1
## S1M-240.2 -1  0  1
##        -1 20 11 56
##        0  11 13  8
##        1  11  4  0
```

Treating the categorical values {“Nay”,“Abstain”,“Aye”} as numeric, {-1, 0, 1}, the scatterplot may be used.



Intuition suggests that it would be better to use *all* of the ballots, rather than just two. But is it possible to first look at the data in search for potential clusters?

- Principal component analysis (PCA) is a potential solution.
  - Recall: PCA creates new variables (as linear functions of old variables), ordered by its importance.
  - New variables are called principal components.

1. Estimate principal components from the data;

```
pc.Votes <- wide.Votes %>% select(-name) %>% prcomp()
```

2. Take the first two principal components;

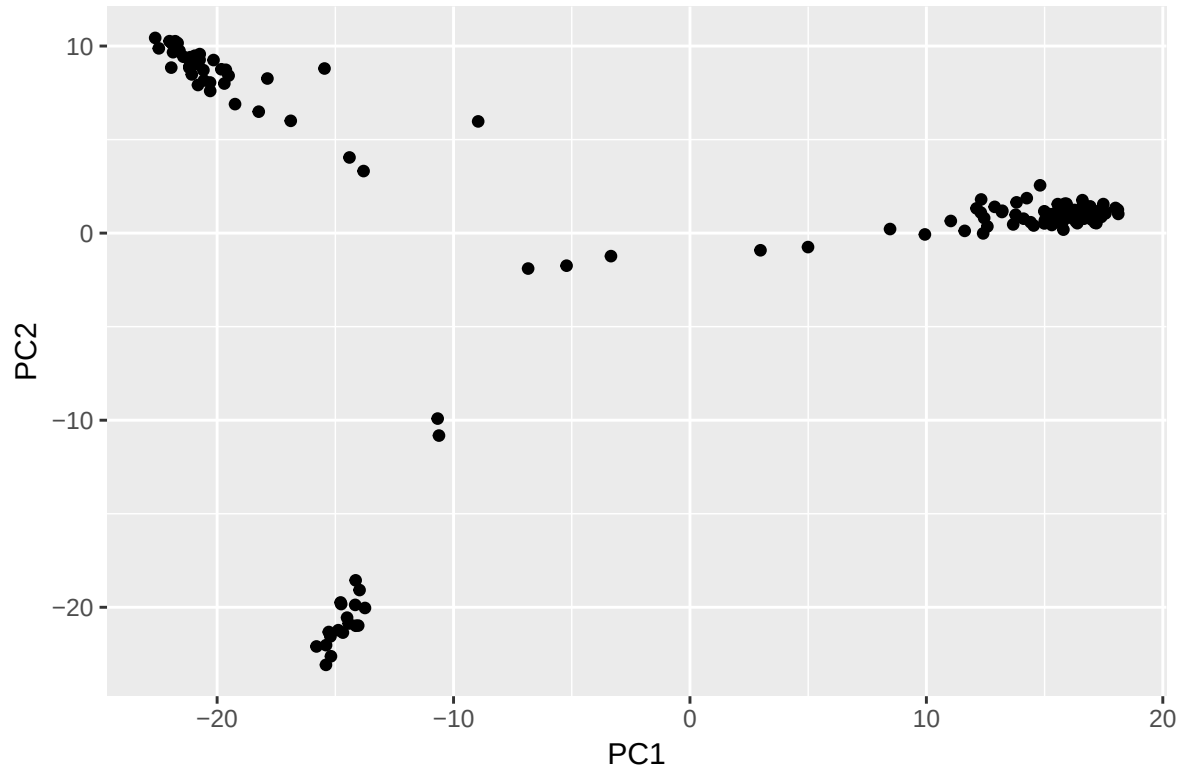
```
pc.Votes$x %>% as_tibble() %>%
  bind_cols(wide.Votes) %>% select(name,PC1,PC2) %>% head(3
```

```
## # A tibble: 3 x 3
##                     name       PC1         PC2
##                    <chr>     <dbl>       <dbl>
## 1           Adam, Brian -21.86319   9.670291
## 2         Aitken, Bill -15.18436 -22.619002
## 3 Alexander, Ms Wendy  13.19690   1.129474
```

- `pc.Votes$rotation` contains the coefficients in computing the values of the new variables.
- `pc.Votes$x` contains the the values of the new variables.

3. Visually inspect the two variables.

```
pc.Votes$x %>% as_tibble() %>% select(PC1,PC2) %>%
  bind_cols(wide.Votes) %>%
  ggplot(aes(x = `PC1`, y = `PC2`)) + geom_point()
```



Principal components are used in both visualization of the data and analysis result (= clustering result).

```
pc.cluster.Votes <- pc.Votes$x %>% as_tibble() %>%
  mutate(cluster = factor(kmeans(x = ., centers = 3)$cluste

pc.cluster.Votes %>%
  select(PC1,PC2,cluster) %>%
  ggplot(aes(x = PC1, y = PC2, color = cluster)) + geom_poi
```